Text Analysis Course

Neural Nets for Texts

Ashuha Arseniy

Moscow Institute of Physics and Technology
Department of Innovation and High Technology

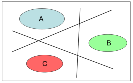
ars.ashuha@gmail.com

May 11, 2016

The task is to assign a document $\boldsymbol{x}$ to one or more classes or categories $\boldsymbol{y}$

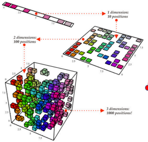
- Linear Models ($\hat{y}$ – scores vector)

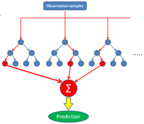
$$\hat{y}(\boldsymbol{x}; W, \boldsymbol{b}) = W\boldsymbol{x} + \boldsymbol{b}$$



- Reduce Dimensions and Ensembles

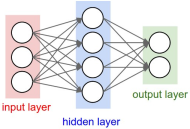
$$\hat{y}(\boldsymbol{x}; T) = \frac{1}{|T|} \sum_i T_i(low\_dim(\boldsymbol{x}))$$



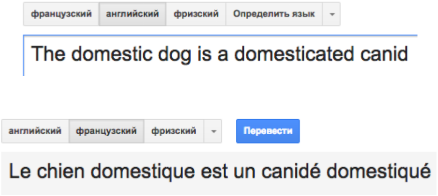


- Neural Nets

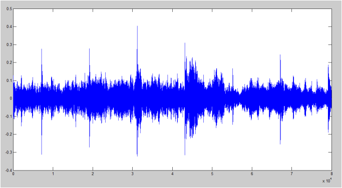
$$\hat{y}(\boldsymbol{x}; W, \boldsymbol{b}) = W^2\sigma(W^1\boldsymbol{x} + \boldsymbol{b}^1) + \boldsymbol{b}^2$$



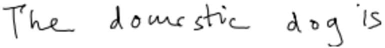
How to 1) take into account the words range 2) make features automatic



(a) Machine Translation



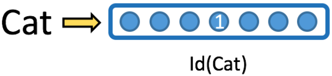
(b) Speech Recognition



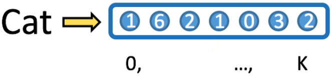
(c) Handwritten Generation

(d) Image Annotation

- Let's back to Text Classification
- Bag of Words is really bad idea, we need to save sequence structure
- Map each word to vector (Embedding)

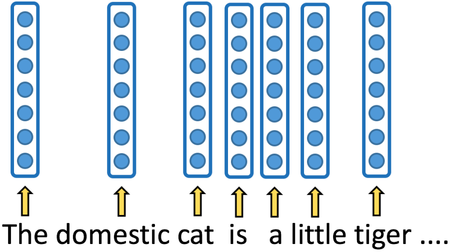


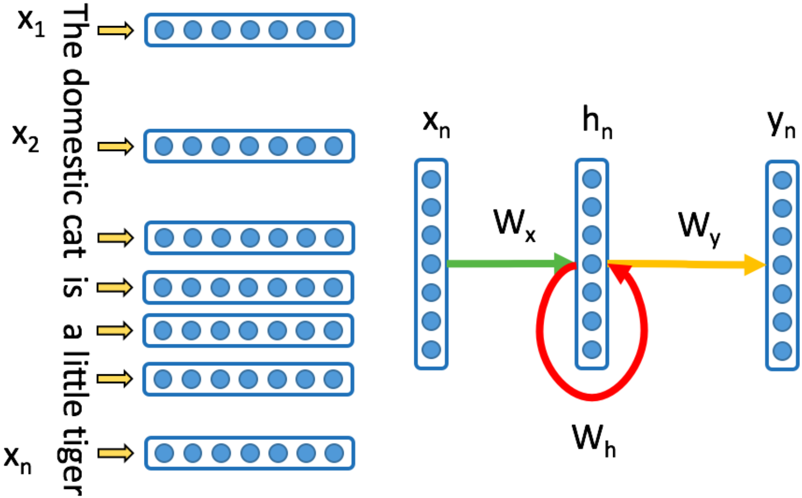
One Hot Encoding



Word2Vec

- Represent Text as a vector sequence





$$h_n = W_x x_n + W_h \sigma(h_{n-1})$$

$$y_n = \sigma(W_y h_n)$$

$$\hat{y}(\boldsymbol{x}; W, \boldsymbol{b}) = W\boldsymbol{x} + \boldsymbol{b}$$

- Introduce and optimize loss function

$$L(X, y, W, b) = \sum_i ||\hat{y}(\boldsymbol{x}_i; W, \boldsymbol{b}) - y_i||^2 \rightarrow \min_{W,b}$$

- Evaluate Gradients The gradient is matrix/vector construct from

$$\frac{\partial L}{\partial W_{ij}} = \ldots \quad \frac{\partial L}{\partial b_i} = \ldots$$

- Run you favourite optimization Method

$$\hat{y}(\boldsymbol{x}; W, \boldsymbol{b}) = W^2\sigma(W^1\boldsymbol{x} + \boldsymbol{b}^1) + \boldsymbol{b}^2$$

$$f^1(x) = \sigma(W^1\boldsymbol{x} + \boldsymbol{b}^1); \qquad f^2(x) = W^2 f^1(x) + \boldsymbol{b}^2$$

- Evaluate Gradients
  - Chain Rule

$$\frac{\partial L(g(x))}{\partial x} = \frac{\partial L}{\partial g}\frac{\partial g}{\partial x} \qquad \frac{\partial L(\boldsymbol{g}(\boldsymbol{x}))}{\partial x_i} = \sum_k \frac{\partial L}{\partial \boldsymbol{g}_k}\frac{\partial \boldsymbol{g}_k}{\partial x_i}$$

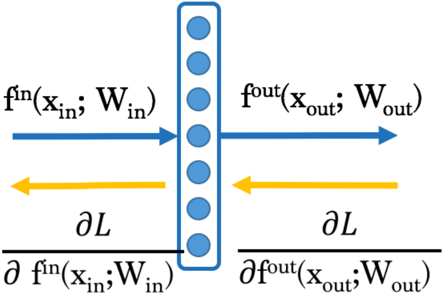
  - Derivation by Weights (Scalar by Scalar is qe Matrix Multiplication)

$$\frac{\partial L}{\partial W^1_{ij}} = \sum_k \frac{\partial L}{\partial f^1_k}\frac{\partial f^1_k}{W^1_{ij}} = \frac{\partial L}{\partial f^1}^T \frac{\partial f^1}{W^1}$$

$\frac{\partial L}{\partial f^1}$ – partial Loss wrt layer output $\qquad \frac{\partial f^1}{W^1}$ – partial layer wrt parameters

  - Get partial loss by output from previous layer

$$\frac{\partial L}{\partial f^1} = \frac{\partial L}{\partial f^2}\frac{\partial f^2}{\partial f^1}$$



- Gradient wrt parameters

$$\frac{\partial L}{\partial W_{out}} = \frac{\partial L}{\partial f^{out}(x_{out}, W_{out})} \frac{\partial f^{out}(x_{out}, W_{out})}{W_{out}}$$

- Gradient wrt input

$$\frac{\partial L}{\partial f^{in}(x_{in}; W_{in})} = \frac{\partial L}{\partial f^{out}(x_{out}, W_{out})} \frac{\partial f^{out}(x_{out}, W_{out})}{f^{in}(x_{in}; W_{in})}$$

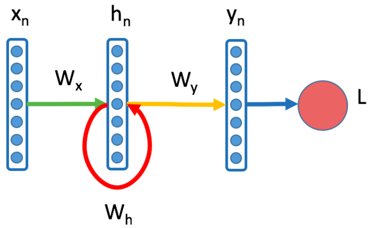
- Affine

```
class AffineLayer():
  def forward(x, params=(w, b)):
    return w.dot(x) + b #compute affine function

  def backward(x, params=(w, b), dout):
    dx = d_x(x, dout) # derivation wrt input
    dw = d_w(w, dout)  # derivation wrt parametrs
    db = d_b(b, dout)  # derivation wrt parametrs
    return (db, dw), dx
```

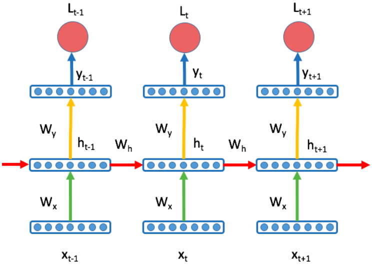
- Sigmoid

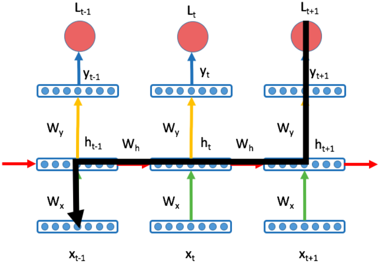
```
class SigmoidLayer():
  def forward(x, params=()):
    return np.sigm(x) #compute sigmoid function

  def backward(x, () dout):
    dx = dx(x, dout) # derivation wrt input
    return (), dx
```

RNN



Unwrapping RNN

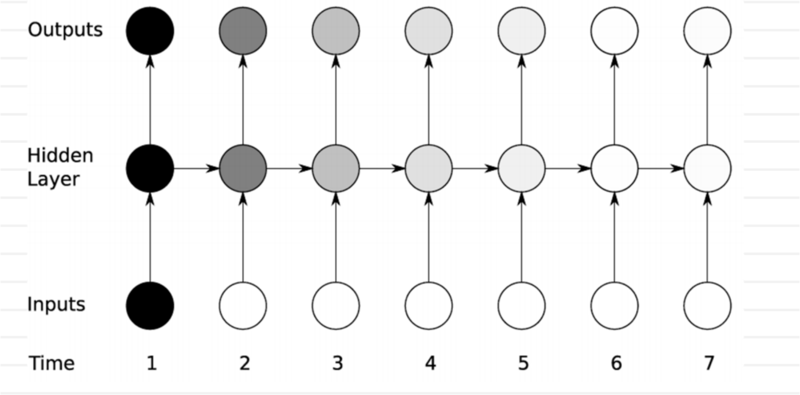


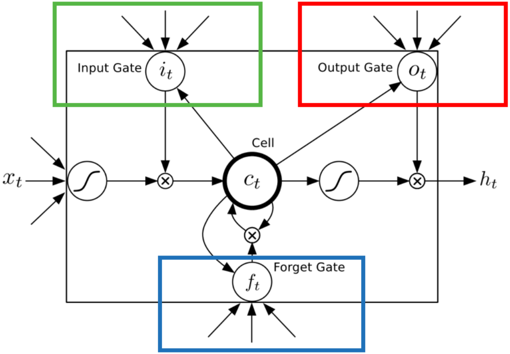


$$E = \sum_{t=1}^{s} E_t; \quad \frac{\partial E}{\partial W_h} = \sum_{t=1}^{s} \frac{\partial E_t}{\partial W_h}; \quad \frac{\partial E_t}{\partial W_h} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_h}$$

$$||\frac{\partial h_t}{\partial h_k}|| < const(W)^{t-k}$$

Gradients **1)** $const(W) > 1$ Boom **2)** $const(W) < 1$ Vanishing


Outputs
Hidden Layer
Inputs
Time
1
2
3
4
5
6
7



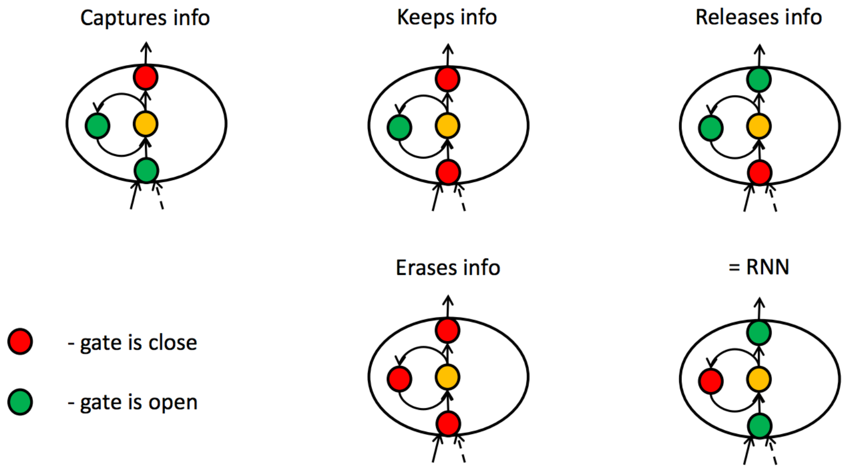
$$y_i = output_i \odot \sigma(h_t)$$

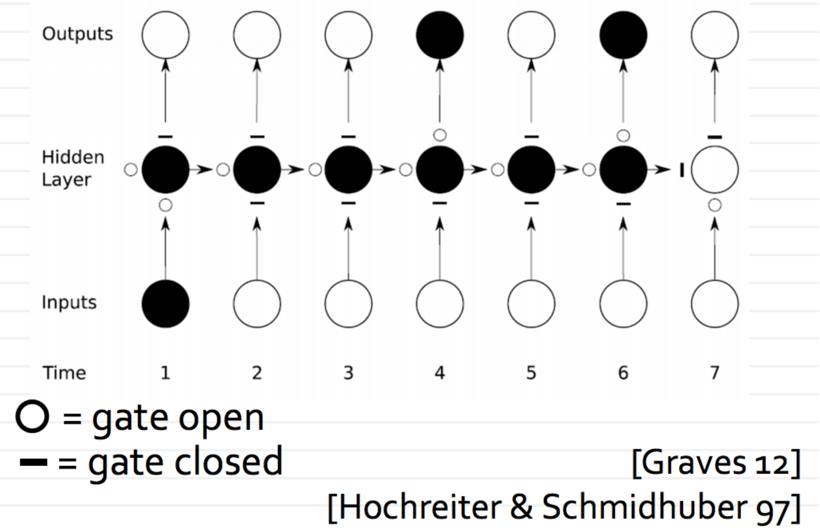
$$h_i = forget_i \odot h_{i-1} + input_i \odot \sigma(W_x x + W_h h_{t-1})$$

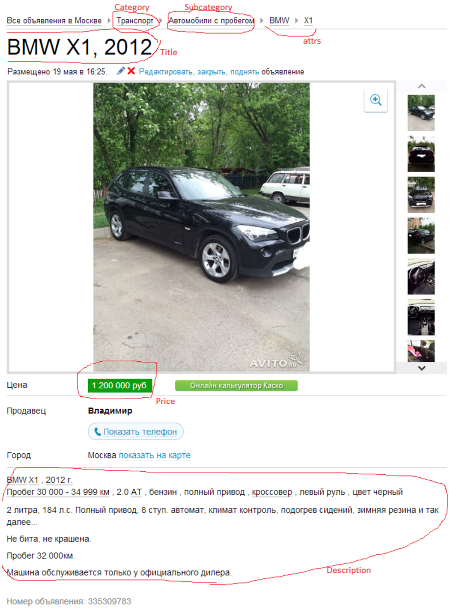
$$input_i = \sigma(W_x^{input} x_t + W_h^{input} h_{t-1} + b^{input})$$

$$forget_i = \sigma(W_x^{forget} x_t + W_h^{forget} h_{t-1} + b^{forget})$$

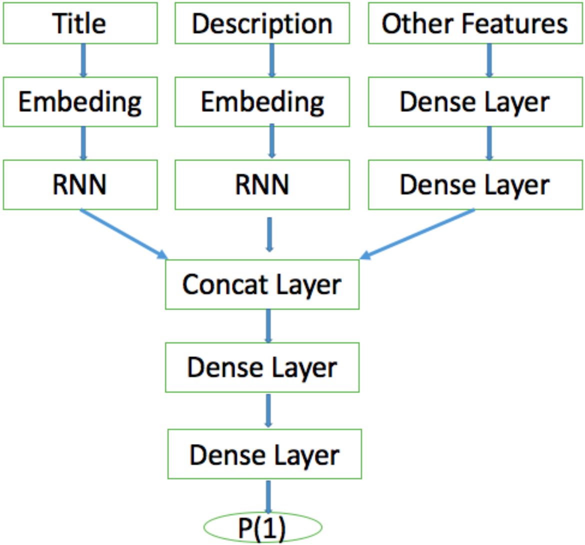
$$output_i = \sigma(W_x^{output} x_t + W_h^{output} h_{t-1} + b^{output})$$


Captures info
Keeps info
Releases info
Erases info
= RNN
- gate is close
- gate is open

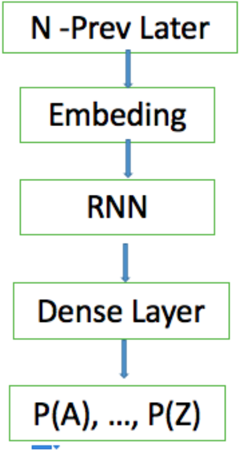

Outputs
Hidden Layer
Inputs
Time
1
2
3
4
5
6
7
O = gate open
— = gate closed
[Graves 12]
[Hochreiter & Schmidhuber 97]



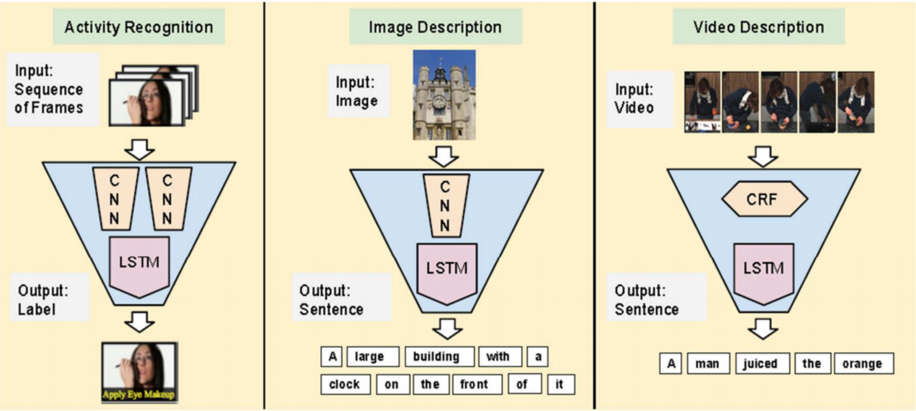
Category
Subcategory
Все объявления в Москве
Транспорт
Автомобили с пробегом
BMW
X1
attrs
BMW X1, 2012
Title
Размещено 19 мая в 16:25.
Редактировать, закрыть, поднять объявление
AVITO.RU
Цена
1 200 000 руб.
Онлайн калькулятор Каско
Price
Продавец
Владимир
Показать телефон
Город
Москва показать на карте
BMW X1 , 2012 г.
Пробег 30 000 - 34 999 км , 2.0 AT , бензин , полный привод , кроссовер , левый руль , цвет чёрный
2 литра, 184 л.с. Полный привод, 8 ступ. автомат, климат контроль, подогрев сидений, зимняя резина и так далее...
Не бита, не крашена.
Пробег 32 000км.
Машина обслуживается только у официального дилера.
Description
Номер объявления: 335309783



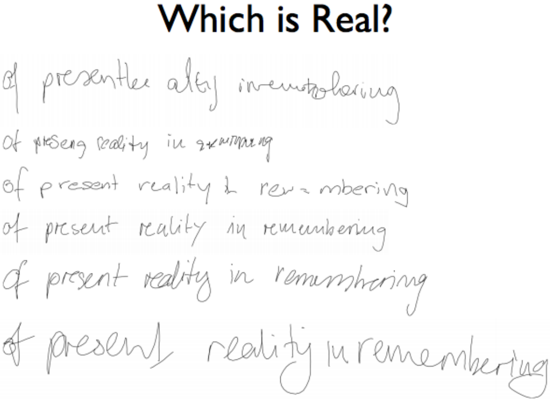
1. Ipython Example

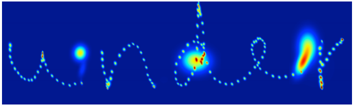


1. Ipython Example
2. http://karpathy.github.io/2015/05/21/rnn-effectiveness/



1. http://cs.stanford.edu/people/karpathy/deepimagesent/generationdemo/





1. http://www.cs.toronto.edu/ graves/handwriting.html
2. http://arxiv.org/abs/1308.0850

Advantages

+ Learn features automatically
+ Complex non-linear Models
+ Save sequence structure

DisAdvantages

– Computational hard
– Hard to optimization (non-convex, ...)
– Available for Over-fit (how can we solve it?)